Vol. 14 No. 4 天 文 研 究 与 技 术 第 14 卷 第 4 期
Oct., 2017 ASTRONOMICAL RESEARCH AND TECHNOLOGY 2017 年 10 月

CN 53-1189/**P** **ISSN** 1672-7673

一种基于 MPI+CUDA 的高性能天文数据分发方法*

方 兵[1]，邓 辉[1]，张晓丽[1]，梅 盈[2]，石聪明[1]，陈晓林[1]，
戴 伟[1,2]，吴静平[1]，王 锋[1,2]

(1. 昆明理工大学云南省计算机技术应用重点实验室，云南 昆明 650500；2. 中国科学院云南天文台，云南 昆明 650011)

摘要：天文海量数据的出现给天文软件的开发带来了诸多挑战。近年来，随着并行计算技术的发展，MPI+GPU 逐渐成为当前高性能天文数据处理的主要模式。针对太阳高分辨图像重建中如何提高重建性能这一问题，对其中的数据读取与数据分发方法进行了系统研究。传统的 MPI 并行处理时，主进程将原图切割成子块，随后将子块发送到各子进程重建，重建后的结果返回主进程。当子进程数量较大且计算节点数量较少时，这种数据分发过程显著增加通讯时长，影响整个重建过程的效率。提出 MPI+CUDA 的一种树状数据分发方法，给出了算法的基本思路与实现方法。实验结果表明，树状分发方式比一般采用的平行分发方式可以提高速度近一倍，成果对天文海量数据开发处理有一定的借鉴作用。

关键词：图像重建；MPI+GPU；数据分发

中图分类号：TP311.1 **文献标识码**：A **文章编号**：1672-7673(2017)04-0481-07

天文海量数据的出现极大地推动了高性能计算技术的发展。其中，图形处理器(Graphics Processing Unit，GPU)作为新一代高密度分布计算体系，成为当前高性能计算的一个亮点。美国 NVIDIA 公司推出的统一计算设备架构(Compute Unified Device Arcthitecture，CUDA)技术，以 C 语言为基础，因其良好的可编程架构和完整的服务成为目前使用最广泛的通用计算图形处理器(General Purpose GPU，GPGPU)模型[1]。

消息传递接口(Message Passing Interface，MPI)是最常用的高性能计算支撑环境[2]，结合 MPI 与 CUDA 技术的优点，实现 MPI 与 CUDA 结合的混合编程模型是一种海量科学数据处理的主流模式。这种混合编程模型中，MPI 负责将任务并行分发，CUDA 负责并行计算，在大幅提高并行效率的同时，使得超大海量数据的处理成为可能。

天文观测工作中的一个重要难点是如何快速处理每天观测得到的海量图像数据。比起以往的提高存贮硬件的存储量后再慢慢处理，天文工作者更加希望能够实时处理观测得到的天文图像，从而减少内存开销，提高整个天文观测工作的效率。随着图形处理器的快速发展以及 MPI+GPU 模型的逐渐成熟与应用，天文工作者逐渐尝试使用 MPI+GPU 的高性能并行模式解决天文海量数据处理的难题。

本文以开发高性能天文软件时遇到的数据分发问题为研究背景，系统研究了基于 MPI+GPU 模型的天文海量数据处理过程中的数据分发方法，结果可为天文海量数据处理过程提供一种新思路。

1 MPI+CUDA 混合编程模型概述

MPI+CUDA 混合编程模型充分结合了 MPI 的粗粒度并行与 CUDA 的细粒度并行，具有高度的并行效率，也是目前主流的并行计算模型。文[3-4]使用这种混合编程模型，测试了大规模矩阵乘问题的并行计算能力，实验结果证明了该模型能够显著提升并行效率。一种简易的 MPI+CUDA 混合模型如图 1。

* 基金项目：国家自然科学基金（U1531132，U1631129，11403009）资助.
收稿日期：2017-01-17；修订日期：2017-02-28
作者简介：方 兵，男，硕士研究生. 研究方向：计算机应用技术. Email：fangbing@cnlab.net
通讯作者：邓 辉，女，教授. 研究方向：分布式计算与科学数据处理、云计算. Email：dh@cnlab.net

从图 1 可以看出，在进程间通信时是中央处理器到图形处理器之间的数据传输，而不是传统的将数据在各节点的中央处理器之间传输，再将数据从中央处理器拷贝到图形处理器进行计算。这是因为使用了英伟达公司的 CUDA-aware MPI① 技术，该技术实现了对图形处理器缓存的直接访问。CUDA-aware MPI 技术可以很好地避免在使用 MPI + CUDA 并行计算时中央处理器与图形处理器之间数据拷贝花费大量时间，从而进一步提高 MPI + CUDA 的并行效率，降低了编程的复杂度，这也使得 MPI + GPU 的模型得到进一步的认可和应用。

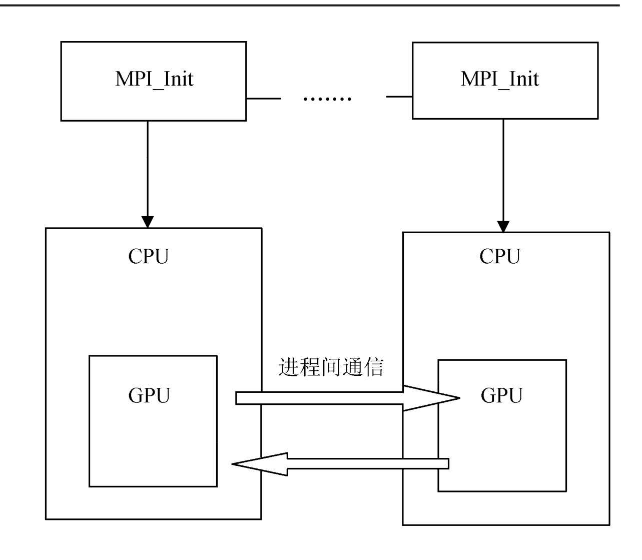


图 1 MPI + CUDA 混合模型示意图

Fig. 1 A diagram of hybrid MPI + CUDA model

MPI + CUDA 的混合编程模型虽然可以取得很好的加速效果，但为了能够进一步提高系统运行速度，一方面可以扩展硬件环境，建立更大的并行集群进行计算；另一方面通过改进 MPI + CUDA 的通信方式，减少通信开销，从而提高系统的运行速度。但根据 Amdahl 定律[5]：处理器数目的无限增大，并行系统所能达到的加速比上限为 $1/S(S = 1/(1 - a + a/n)$)，其中，a 为并行计算部分所占比例，n 为并行处理节点个数。可以看出并行加速不仅受限于程序的串行分量，而且也受限于并行程序运行时的额外开销。一个更大的并行集群由于通信时长等因素，不一定能取得更理想的效果。因此，对 MPI + CUDA 混合编程模型的研究有利于为该模式寻找一种更高效的通信方式，从而进一步提高 MPI + CUDA 混合编程模型的效率。

2 MPI 与 CUDA 在天文数据处理的相关工作

随着消息传递接口的广泛应用与图形处理器的快速发展，消息传递接口和 CUDA 在天文图像数据处理方向的应用取得了一定的进展。文[6]在 CUDA 环境下实现了一个 Hα 全日面云污染实时识别和修复系统。文[7]利用图形处理器对 Mark5B 数据进行处理，对以 8 MB/s 的实时采样，可以在 0.51 s 内处理完成，实现了对脉冲星观测数据实时处理的要求。文[8]实现了基于图形处理器并行计算的 Level1 级重建程序，使程序在不改变原有处理方式的情况下将重建速度提高近 30 倍，达到了准实时重建的效果。文[9]利用图形处理器实现基于相位差异(Phase Diversity，PD)的地基望远镜图像恢复技术，发挥了高性能计算的优势，取得了数十倍的加速效果。文[10]在研究了太阳高分辨图像重建算法在图形处理器上的实现，为天文数据处理算法在图形处理器上的并行实现提供了方向。文[11]利用高性能集群与消息传递接口技术加速选帧位移叠加(Level1)重建太阳高分辨图像，相比以前的 IDL，实现了 23 倍显著加速，并提出了基于 CUDA 的子块图像重建新的算法，给天文图像数据处理提供了很多新的并行实现方法，展望了 MPI + CUDA 混合编程方法在天文图像重建上的可能应用。

以上研究为 MPI + CUDA 模式在天文海量数据处理方向的应用提供了很好的支持。但这些基本集中于并行方法的研究，对其中的数据分发还没有进行深入的讨论。部分研究工作已经发现数据分发比较耗时，从而影响系统的运行效率。

3 一种基于 MPI + CUDA 的数据分发方法

在高分辨图像重建过程中，需要将原图切割成子图，然后对各个子图进行图像重建算法的处理，

① https://devblogs.nvidia.com/parallelforall/introduction-cuda-aware-mpi/

最后将处理后的子图像拼接整合。通常的做法是使用 MPI 主进程在中央处理器上将原图切割成子块，然后将子块发送到各子进程对应的图形处理器上重建。当多个进程对同一个原图进行取数据时会出现等待的情况，从而影响效率。

为了提高重建效率，提出了在计算节点数量较少时可以采用一种树状 MPI + GPU 的数据分发模型，基本思想是将原始数据分割到各个图形处理器内存，再通过图形处理器向下一级子进程进行数据的切割和发送，图像处理子进程向它们的上一级的多个进程获取数据，进而增加了数据切割分发的并行度，减少了通信等待时间，而且由于图像数据的分发和处理过程都在图形处理器上执行，从而更进一步提高了处理速度。

为验证对比，分别以图像重建算法在图形处理器上实现为基础，并利用对大矩阵的一系列处理过程代替图像算法的重建过程，从而只讨论数据分发过程的区别带来的效率差异。

3.1　实验设计

3.1.1　平行分发 MPI + GPU 算法设计

在 MPI + GPU 图像重建过程中，主进程根据进程数循环将图像切割，然后发送到各个子进程对应的图形处理器进行计算(使用 CUDA-Aware MPI 机制直接从中央处理器内存发送到图形处理器内存)。分发方法如图 2。详细过程：主进程(0 号进程)根据子进程号(假设是 1 号进程)切割一块子图发送给 1 进程所分配的图形处理器(假设是 GPU1)进行运算。运算完成后 1 号进程将结果发送到主进程，并判断原图是否有剩余的图像没有切割完成，如果有，继续为 1 号进程切割子图重建，各个子进程依此方式轮番切割原图进行运算，直到原图切割重建完毕。

3.1.2　树状 MPI + GPU 数据分发设计

树状 MPI + GPU 数据分发，顾名思义是以一种树的方式层层发送接收。该方法以图形处理器个数为基础，将原始数据分成若干块，然后在图形处理器上进行子块的切割，再发送到其余的子进程，即将切割的过程并行化。分发方法如图 3。详细过程：主进程(0 号进程)根据图形处理器个数(假设为 2)将原图切割为 2 等份发送到 1～2 号进程对应的图形处理器内存(使用 CUDA-Aware MPI 机制)。重建子进程(假设 3 号进程)在为它分配的图形处理器(假设为 GPU0)上切割相应子图进行重建，直到各个图形处理器上的图像(1～2 号进程对应的图形处理器)切割重建完毕，然后 1～2 号进程将它们的子进程处理后的结果整合，分别发送到 0 号进程整合。

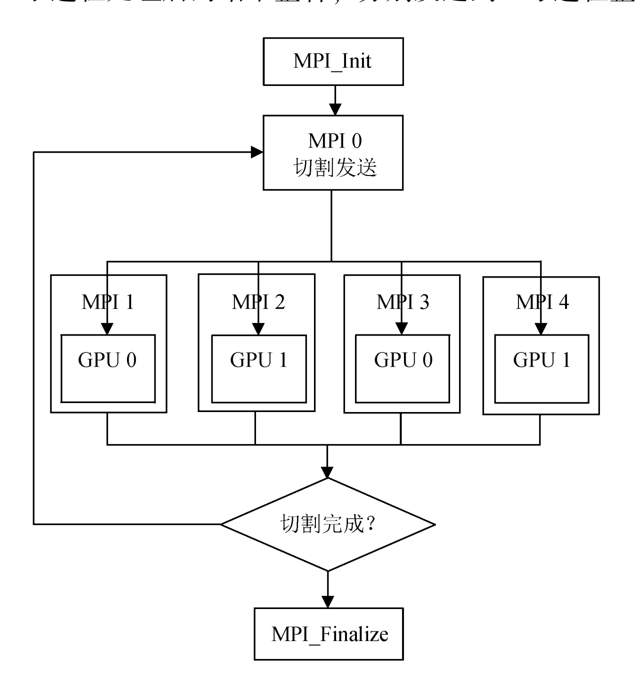


图 2　平行分发 MPI + GPU 算法示意图

Fig. 2　A diagram of flat distribution method

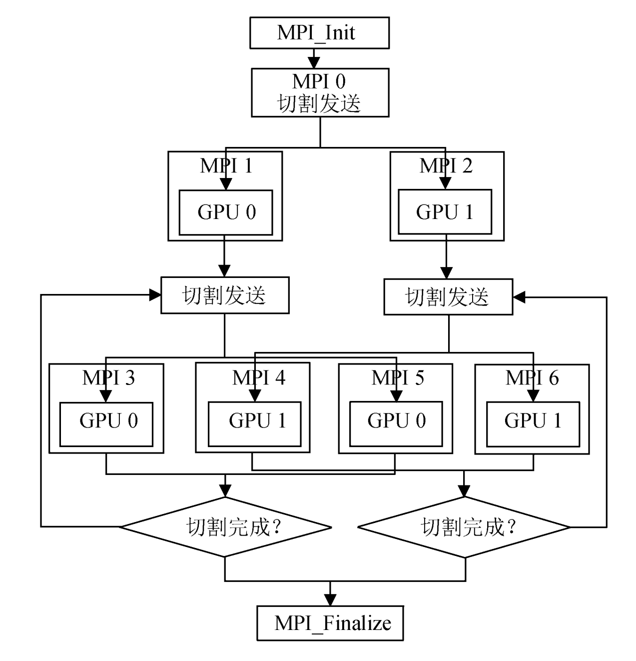


图 3　树状 MPI + GPU 数据分发示意图

Fig. 3　A diagram of MPI + GPU hierarchy distribution

3.2　实验及结果分析

3.2.1　实验环境

实验环境包括两台双 6 核的中央处理器服务器，安装有 2 个双核的 K80 图形处理器，操作系统为 Ubuntu 14.04.4 LTS，安装 OpenMPI2.0.0，CUDA-7.5 环境。

3.2.2　算法实现

根据以上实验设计，实现两种分发方法，其示意代码如表 1。

表 1　两种分发方法示意代码

Table 1　Illustrate code of two distribution methods

平发 MPI + GPU

```
int main(){
//初始矩阵
  init_Matrix();
  if(rank==0){
    //根据进程数为各子进程切割分发子块
    for(i=1;i<usersize;i++){
      subfield();
      MPI_Send();
    }
    //回收子进程处理好的子块，整合；判断原图是否切割完成，否则继续切割分发子块
    while(jobs_done<jobs){
      MPI_Recv();
      combine();
      subfield();
      MPI_Send();
      jobs_done++;
    }
      //回收各子进程最后一次分配的子块，整合
      while(i<usesize){
      MPI_Recv();
      combine();
    }
  } else{
    //子进程接收主进程发送的子块，对子块进行处理后发送回主进程
    MPI_Recv();
    add_Kernel();
    MPI_send
  }
    MPI_Finalize();
  }
```

树状 MPI + GPU

```
int main(){
//初始矩阵
  init_Matrix();
  if(rank==0){
    //主进程根据 GPU 数将原图切割发送到各个 2 级子进程对应的 GPU
    for(i=0;i<devCount;i++){
      subfield();
      MPI_Send();
    }
```

（续表 1）

树状 MPI + GPU

```
    //回收各二级子进程整合好的图像,进行最终整合
    while(i<devCount){
      MPI_Recv();
      combine();
    }
    }
    //二级子进程将 GPU 上的数据切割发送到三级子进程对应的 GPU
    int tid=rank%devCount
    if(0<rank&&rank<devCount+1){
      MPI_Recv();
      for(i=devCount+1;i<user_size;i++){
        if(i%devCount==tid){
        subfield();
     MPI_Send();
      }
    }
    //回收三级子进程处理好的子块,整合;判断二级子进程对应的 GPU 上的数据是否切割完成,否则继续切割分发子块
    while(jobs_done<jobs){
      MPI_Recv();
      combine();
      subfield();
      MPI_Send();
      jobs_done++;
    }
    //回收各三级子进程最后一次分配的子块,整合
    while(i<j){
      MPI_Recv();
      combine();
    }
    //二级子进程将整合好的图像数据发送给一级主进程
    MPI_Send();
  }else{
    //三级子进程接收二级子进程发送的子块,对子块进行处理后发送回二级子进程
    MPI_Recv();
    add_Kernel();
    MPI_Send();
  }
    MPI_Finalize();
  }
```

3.2.3　实验及结果分析

由于实验环境安装有两个双核的 K80 图形处理器，相当于有 4 个单核的图形处理器，因此在实验过程中能读到 4 个图形处理器，并同时使用这 4 个图形处理器进行运算。在实验中对初始化的大矩阵进行处理。通过对矩阵的切割分发实现高分辨图像重建过程中图像的切割和子图的分发；通过对矩阵的耗时处理实现图像重建过程中对子图的一系列图像算法的处理。实验分别处理了 10 240 × 10 240 的大矩阵和 20 480 × 20 480 的大矩阵，切割的子块大小分别为 1 024 × 1 024 和 2 048 × 2 048，然后在图形处理器上对切割的子块进行耗时处理，当子进程对数据子块的处理结束后将结果发送到上一级进程整合。通过调用不同的进程数和处理不同大小的数据块得到如图 4 的实验结果。

图 4 为在一台服务器上对 10 240 × 10 240 矩阵的处理结果，图 5 为在一台服务器上对 20 480 × 20 480 矩阵处理的结果。由图 4 可以看出，随着进程数和通信开销的增加，处理时间也随之增加。树发方式虽然也呈上升趋势但总体要优于平发方式。由图 5 可以看出，平发随着进程数的增加，呈现一个凹型曲线。这是由于随着处理进程的增加，当进程数增加到一定的程度其处理效率增加(如增加到 15 个进程)，但超出之后效率就会由于通信时间的增加而变低。树发方式还是比平发方式有大概 1 倍的性能提升。

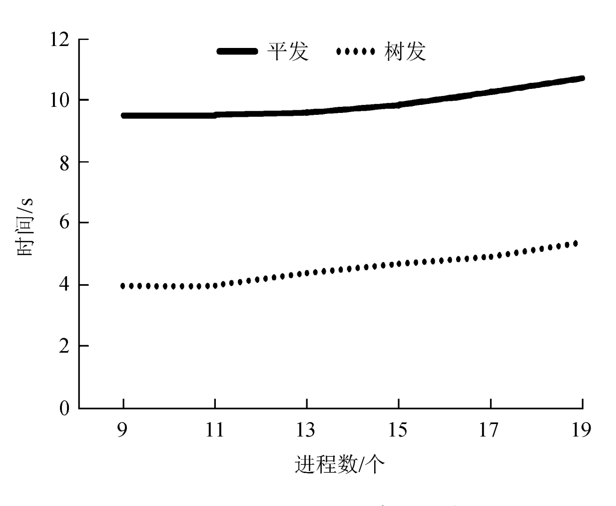


图 4　10 240 × 10 240 矩阵处理结果

Fig. 4　Processing results of matrix with 10240 × 10240

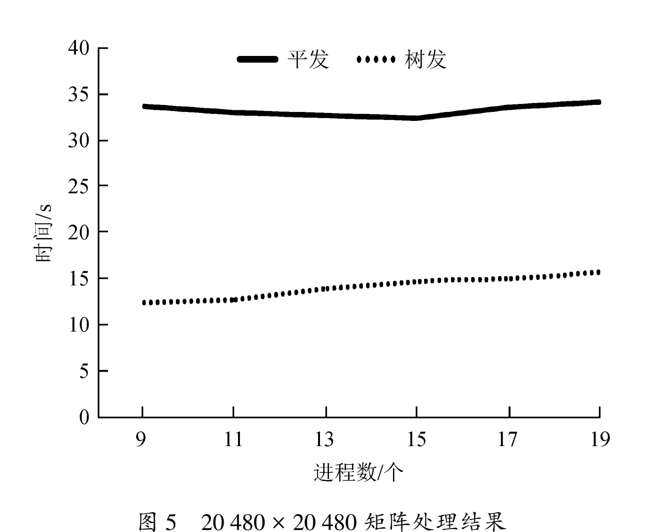


图 5　20 480 × 20 480 矩阵处理结果

Fig. 5　Processing results of matrix with 20480 × 20480

为了体现在集群上的数据分发效果，对 20 480 × 20 480 矩阵在两台服务器上进行处理，其中二级进程数为 4，得到了如图 6 的实验结果。由图 6 的结果可以看出，在集群环境下树发方式优于平发方式。图 6 中树发曲线可以看到一个凹型，那是由于当分配的进程数较少时，程序没有充分利用集群中的图形处理器；当进程数较大时，又会出现多个进程获取同一图形处理器时出现阻塞的问题。

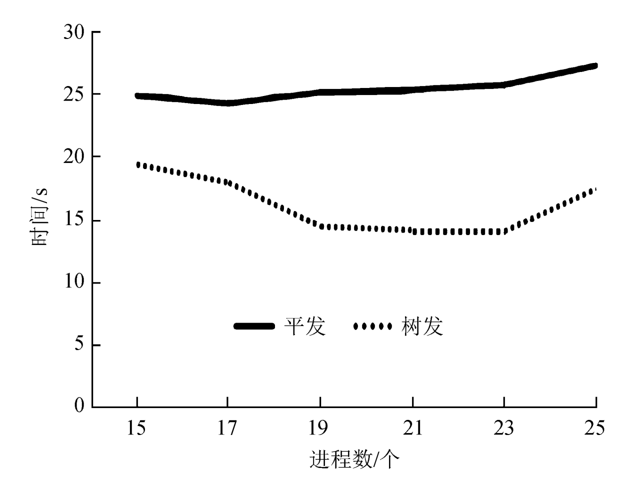


图 6　集群中 20 480 × 20 480 矩阵处理结果

Fig. 6　Processing results of matrix with 20480 × 20480 in the cluster

综上，需要注意的是在进程分配时要考虑到图形处理器个数和中央处理器核数，进程数要大于图形处理器个数才能充分利用现有图形处理器，最好是大于图形处理器个数的两倍。因为在树状数据分发下，一般需要有与图形处理器个

数相同的进程作为控制进程来控制与第 3 级子进程的数据分发与接收合并，如果二级进程数少于图形处理器数将导致数据在图形处理器之间来回发送，在集群环境下还要考虑大于中央处理器核数。二级子进程的数目也不宜过多，因为考虑到图像数据处理的特性，过分地拆分原图大小再进行重建，合并后的图像数据误差可能增大，并且二级进程的数目也是控制进程数目，因此需要合理地控制二级进程的数目，使得三级数据处理进程大于控制进程。

4 结束语

本文重点讨论了并行计算中的数据分发机制，实验比较得出树状的 MPI + GPU 分发方式相比平面分发的方式有较好的优势，这为高分辨图像重建过程中的数据分发方式提供了一个很好的参考。

除 MPI + CUDA 混合模型受到广泛研究与应用外，MPI + openMP 混合编程模式由于能充分利用共享存储模型和消息传递模型的优点，在很多领域也被广泛应用。因此，MPI + openMP + CUDA 混合模型的性能和可行性逐渐开始被研究，在天文海量数据处理上的应用将是下一步研究的内容。

参考文献：

[1] Luebke D. CUDA：scalable parallel programming for high-performance scientific computing [C] // 5th IEEE International Symposium on Biomedical Imaging：From Nano to Macro. 2008：836-838.

[2] Gropp W，Lusk E，Doss N，et al. A high-performance，portable implementation of the MPI message-passing interface standard [J]. Parallel Computing，1996，22(6)：789-828.

[3] 许彦芹，陈庆奎. 基于 SMP 集群的 MPI + CUDA 模型的研究与实现 [J]. 计算机工程与设计，2010，31(15)：3408-3412.
Xu Yanqin，Chen Qingkui. Research and implementation of MPI + CUDA model based on SMP clusters [J]. Computer Engineering and Design，2010，31(15)：3408-3412.

[4] 刘青昆，马名威，阎慰椿. 基于 MPI + CUDA 异步模型的并行矩阵乘法 [J]. 计算机应用，2011，31(12)：3327-3330.
Liu Qingkun，Ma Mingwei，Yan Weichun. Parallel matrix multiplication based on MPI + CUDA asynchronous model [J]. Journal of Computer Applications，2011，31(12)：3327-3330.

[5] Amdahl G M. Validity of the single processor approach to achieving large scale computing capabilities [C] // AFIPS Spring Joint Computer Conference. 1967.

[6] 张能维，杨云飞，李冉阳，等. Hα 全日面云污染实时识别和修复系统 [J]. 天文研究与技术，2016，13(2)：242-249.
Zhang Nengwei，Yang Yunfei，Li Ranyang，et al. A real-time image processing system for detecting and removing cloud shadows on Hα full-disk solar [J]. Astronomical Research & Technology，2016，13(2)：242-249.

[7] 李佳功，徐永华，李志玄，等. 基于 Mark5B + GPU 脉冲星观测系统 [J]. 天文研究与技术——国家天文台台刊，2014，11(4)：335-342.
Li Jiagong，Xu Yonghua，Li Zhixuan，et al. An observation system for pulsars based on a GPU architecture and a Mark5B [J]. Astronomical Research & Technology——Publications of National Astronomical Observatories of China，2014，11(4)：335-342.

[8] 施正，向永源，邓辉，等. 一米真空太阳望远镜 Level 1 级图像选帧的 GPU 实现 [J]. 科学通报，2015，60(15)：1408-1413.

Shi Zheng, Xiang Yongyuan, Deng Hui, et al. A method of Level 1 frames-selection based on GPU for new vacuum solar telescope [J]. Science Bulletin, 2015, 60(15): 1408-1413.

[9] 张楠. 基于相位差异的地基望远镜图像恢复算法与 GPU 高速实现 [D]. 长春：中国科学院长春光学精密机械与物理研究所, 2012.

[10] 向永源. 太阳高分辨高速重建算法的研究 [D]. 昆明：中国科学院云南天文台, 2016.

[11] 李雪宝. 太阳望远镜海量数据并行处理技术研究 [D]. 昆明：中国科学院云南天文台, 2015.

A High-Performance Distribution Method of Astronomical Data Based on MPI + CUDA

Fang Bing[1], Deng Hui[1], Zhang Xiaoli[1], Mei Ying[2], Shi Congming[1], Chen Xiaolin[1], Dai Wei[1,2], Wu Jingping[1], Wang Feng[1,2]

(1. Key Laboratory of Computer Technology Application, Kunming University of Science and Technology, Kunming 650500, China, Email: dh@cnlab.net; 2. Yunnan Observatories, Chinese Academy of Sciences, Kunming 650011, China)

Abstract: The appearance of massive astronomical data has brought a lot of challenges to the development of astronomy software. In recent years, with the development of parallel computing technology, MPI + GPU mode has become the main mode for current high performance astronomical data processing gradually. For the problem of how to improve reconstruction performance in reconstruction of solar high resolution image, this paper has made systematic research on data reading and data distribution method. During traditional MPI parallel processing, master process cuts original image into sub-blocks, and then delivers sub-blocks into each sub-process for reconstruction, the results after reconstruction will be returned to the master process. When the number of sub-process is big and calculation nodes are few, this data distribution process will increase the time for communication significantly and affect the efficiency of the whole reconstruction process. This paper has proposed a tree data distribution method under MPI + CUDA and offered basic ideas and realization method of the algorithm. Experimental results have shown that tree distribution mode nearly doubles the speed than generally adopted plane distribution mode; its achievements provide certain reference for the development and processing of massive astronomical data.

Key words: Image reconstruction; The MPI + GPU; Data distribution